REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 26-06-2014	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* 1 Jun 2012 – 31 May 2014

4. TITLE AND SUBTITLE Moving horizon estimation for navigation application	**5a. CONTRACT NUMBER** FA2386-12-1-4004
	5b. GRANT NUMBER Grant AOARD-124004
	5c. PROGRAM ELEMENT NUMBER 61102F
6. AUTHOR(S) Prof. Keck Voon Ling	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Nanyang Technological University, Nanyang Ave, Nanyang 639798, Singapore	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AOARD, UNIT 45002, APO AP 96338-5002	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/AFOSR/IOA(AOARD)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) AOARD-124004

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution A: Approved for public release. Distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This work investigated the feasibility of solving Moving Horizon Estimation(MHE) problems in hardware, e.g. Field Programmable Gate Arrays(FPGA), in contrast to using software. With the hardware approach we could build customized computational machines for specific algorithms and hence gain computational and power efficiency, which is crucial for embedded real-time applications where computational resources are limited. A hardware approach enables one to build certifiable components and is also less susceptible to software virus attack. Using high level synthesis tools instead of low level hardware description languages, we demonstrated that it is feasible to routinely deploy customized, sophisticated computational algorithms such as MHE, on reconfigurable hardware for real-time embedded applications.

15. SUBJECT TERMS

Navigation,Guidance, and Control, Micro Air Vehicles (MAVs)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Brian Sells, Lt Col, USAF
U	U	U	SAR	28	**19b. TELEPHONE NUMBER** *(Include area code)* +81-3-5410-4409

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **26 JUN 2014**	2. REPORT TYPE **Final**	3. DATES COVERED **01-06-2012 to 31-05-2014**

4. TITLE AND SUBTITLE **Moving horizon estimation for navigation application**	5a. CONTRACT NUMBER **FA2386-12-1-4004**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER **61102F**
6. AUTHOR(S) **Keck Voon Ling**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Nanyang Technological University,Nanyang Ave,Nanyang 639798,Singapore,SG,639798**	8. PERFORMING ORGANIZATION REPORT NUMBER **N/A**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **AOARD, UNIT 45002, APO, AP, 96338-5002**	10. SPONSOR/MONITOR'S ACRONYM(S) **AFRL/AFOSR/IOA(AOARD)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) **AOARD-124004**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This work investigated the feasibility of solving Moving Horizon Estimation(MHE) problems in hardware e.g. Field Programmable Gate Arrays(FPGA), in contrast to using software. With the hardware approach we could build customized computational machines for specific algorithms and hence gain computational and power efficiency, which is crucial for embedded real-time applications where computational resources are limited. A hardware approach enables one to build certifiable components and is also less susceptible to software virus attack. Using high level synthesis tools instead of low level hardware description languages, we demonstrated that it is feasible to routinely deploy customized, sophisticated computational algorithms such as MHE, on reconfigurable hardware for real-time embedded applications.

15. SUBJECT TERMS
Navigation,Guidance, and Control, Micro Air Vehicles (MAVs)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**28**	

Final Report for AOARD Grant 124004

Moving Horizon Estimation on a Chip

K. V. Ling (ekvling@ntu.edu.sg)
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
Phone: +65 6790 5567 Fax: +65 6793 3318

Period of Performance: 1 June 2012 - 31 May 2014

Abstract

This work investigated the feasibility of solving Moving Horizon Estimation(MHE) problems in hardware, e.g. Field Programmable Gate Arrays(FPGA), in contrast to using software. With the hardware approach we could build customized computational machines for specific algorithms and hence gain computational and power efficiency, which is crucial for embedded real-time applications where computational resources are limited. A hardware approach enables one to build certifiable components and is also less susceptable to software virus attack. Using high level synthesis tools instead of low level hardware description languages, we demonstrated that it is feasible to routinely deploy customised, sophisticated computational algorithms such as MHE, on reconfigurable hardware for real-time embedded applications.

In this report, we first reviewed one formulation of the MHE estimation which is amenable for both linear and non-linear systems. Numerical simulations, including a radar tracking problem, were performed to compare its performance against traditional Kalman Filter and Extended Kalman Filter (EKF). One advantage of the MHE approach is that prior information, such as constraints that exist in applications can be incorporated in the design. The results showed that the incorporation of constraints in the MHE approach outperformed both the Kalman Filter and EKF when there is poor prior knowledge of the process and measurement noises.

MHE is an online optimization-based strategy which can be formulated as a Quadratic Program(QP), which then needs to be solved at every sampling instance. This can be computationally more expensive then the traditional Kalman Filter(KF) where recursive solution is available. We developed various MHE designs and implemented them on the Xilinx Zynq ZC702 FPGA board. The results suggest that, for modest size MHE problems, a hardware solution could outperform software solution, with very attractive speed, clock and power efficiency. Instead of low level hardware description language, we used a high-level synthesis tool (Vivado from Xilinx) for prototyping so that the design can be easily customized for different applications and for design space explorations, e.g. trading-off power, hardware resource and computational speed. We concluded that it is feasible to routinely deploy the "MHE on a chip" technology for modest size MHE problems in embedded and real-time applications where power and computational resources are limited.

1 Introduction

The aim of this work is to investigate the feasibility of solving Moving Horizon Estimation(MHE) problems in hardware, e.g. Field Programmable Gate Arrays(FPGA), in contrast to using software. With the hardware approach we could build customized computational machines for specific algorithms and hence gain computational and power efficiency, which is crucial for embedded real-time applications where computational resources are limited. A hardware approach enables one to build certifiable components and is also less susceptable to software virus attack. The ultimate aim of this work is to demonstrate that, by using high level synthesis tools instead of low level hardware description languages, it is feasible to routinely deploy customised, sophisticated computational algorithms such as MHE, on reconfigurable hardware for real-time and embedded applications.

One paper was published[1] and another is currently under review[2]. Both papers are included in this report. The appendex contains the C code and photos of the experimental set up.

2 Results and Discussions

2.1 MHE is more robust than KF or EKF against inaccurate noise statistics

MHE is an online optimization-based strategy, unlike Kalman Filter (KF) whose design depends crucially on a good knowledge of the statistical characteristics of the noise in the system. In other words, KF's performance is sensitive to the accurate assumption of the noise statistics. Hence, careful tuning of KF is usually needed when deploying KF in real, practical applications. One advantage of the MHE approach is that prior information, such as constraints that exist in applications can be incorporated in the design.

In the first paper of this report, we reviewed one formulation of the MHE estimation which is amenable for both linear and non-linear systems. Numerical simulations, including a radar tracking problem, were performed to compare its performance against traditional Kalman Filter and Extended Kalman Filter (EKF). The results showed that the MHE approach outperformed both the Kalman Filter and EKF when there is poor prior knowledge of the process and measurement noises. This confirmed our intuition that, by including constraints in the formulation, MHE has the advantage over

[1]D. Zhou, K.V. Ling and E.K. Poh, Constrained Moving Horizon Estimation for Linear and Nonlinear Systems, 3rd SONDRA Workshop, 10–14 June 2013, Hyeres, France, pp.188-191.

[2]T. V. Dang and K.V.Ling, Moving Horizon Estimation on a Chip, submitted to International Conference on Control, Automation, Robotics and Vision (ICARCV), 10-12 December 2014, Singapore

KF and EKF in the sense that the design is less sensitive to wrong choices of Q and R (process and measurement noise variances needed in KF and EKF designs). Robustness of the design against inaccurate prior knowledge (e.g. process and measurement noise variances) is important for practical deployment of the MHE technology.

2.2 Implementation of MHE on FPGA

The next paper demonstrated the feasibility of implementing MHE algorithm on hardware. In a larger context, it demonstrated that, by using high level synthesis tools instead of low level hardware description languages, it is feasible to routinely deploy customised, sophisticated computational algorithms such as MHE, on reconfigurable hardware for real-time and embedded applications.

MHE can be formulated as a Quadratic Program(QP), which then needs to be solved at every sampling instance. Interior Point Method(IPM) and Active Set Method(ASM) are two popular algorithms for solving QP problems. In this work, however, we used Alternating Direction Method of Multipliers(ADMM) which is a first order method. ADMM has the advantage that it has a simpler computational structure and hence more suitable to exploit the computational resources available in FPGA for parallel computation, although ADMM takes more iteration steps to converge than second order methods such as IPM.

Instead of implementing a generic QP solver, we specialized to MHE with box-type constraints (a commonly encountered constraint type in practical applications), so that the ADMM iterations become much simplified and involve mainly dot-product computations This has the potential of achieving a division-less algorithm, leading to fixed-point only computation; division and floating point computations are especially time and resource expensive in hardware. Instead of low level hardware description language, we used a high-level synthesis tool (Vivado from Xilinx) for prototyping so that the design can be easily customized for different applications and for design space explorations, e.g. trading-off power, hardware resource and computational speed.

We developed various MHE designs, compared floating point and fixed point implementations, with and without parallelism (loop pipelining), on a Xilinx Zynq ZC702 FPGA board. These designs clocked at 50MHz. The results suggest that, for modest size MHE problems, a hardware solution could outperform software solution, with very attractive speed, clock and power efficiency. It is thus feasible to deploy the "MHE on a chip" technology for modest size MHE problems in embedded and real-time applications where power and computational resources are limited.

3 List of Publications and Significant Collaborations

Paper published in non-peer-reviewed conference proceedings

D. Zhou, K.V. Ling and E.K. Poh, Constrained Moving Horizon Estimation for Linear and Nonlinear Systems, 3rd SONDRA Workshop, 10–14 June 2013, Hyeres, France, pp.188-191.

Manuscript submitted but not yet published

T. V. Dang and K.V.Ling, Moving Horizon Estimation on a Chip, submitted to International Conference on Control, Automation, Robotics and Vision (ICARCV), 10-12 December 2014, Singapore.

List of interaction and significant collaborations

Significant collaboration with Dr E. K. Poh, DSO National Laboratories, Singapore, which resulted in a project entitled "Mutliband GNSS Software Receiver" valued at approximately SGD324K from Nov 2013–Nov 2016. This project is in partnership with AFRL to develop and test GNSS software receivers which are capable of acquiring and tracking weak navigation signals from the GPS, Glonass, and Compass constellation.

SONDRA

3rd SONDRA WORKSHOP (PROCEEDINGS)

E.M. MODELING, NEW CONCEPTS AND SIGNAL PROCESSING FOR RADAR DETECTION AND REMOTE SENSING

HYERES - LA LONDE LES MAURES (FRANCE)

10-14 JUNE 2013

Conference Board

Marc LESTURGIE	ONERA/SONDRA	William LAU Yuei Khei	DSO
Wee Peng ANG	DSO/SONDRA	Kee Chaing CHUA	NUS
Gilles FLEURY	SUPELEC	Chee Lip GAN	NTU/TL
Jean-Marc BOUTRY	ONERA	Boo Cheong KHOO	NUS/TL
Hian Lim CHAN	DSO		

Organization Board

Anne-Hélène PICOT	SUPELEC/SONDRA	Leonard TAN	DSO

http://www.supelec.fr/sondra

SONDRA is a joint laboratory established between SUPELEC (France), ONERA (The French Aerospace Lab, France), NUS (National University Of Singapore) and DSO National Laboratories

ONERA Supélec NUS National University of Singapore DSO NATIONAL LABORATORIES

Constrained Moving Horizon Estimation for Linear and Nonlinear Systems

Dexiang Zhou, K.V. LING and E.K. Poh

Abstract— **Moving Horizon Estimation (MHE) is formulated as a constrained optimization problem using available measurements over a moving window interval. This approach alleviates the computational complexity and naturally incorporate prior information in the form of inequality constraints on state and noise. In this paper, we review one formulation of the MHE estimation which is amenable for both linear and non-linear systems. Numerical simulations, including a radar tracking problem, are performed to compare its performance against traditional Kalman Filter and Extended Kalman Filter (EKF). The results show that the MHE approach outperforms both filters when there is poor prior knowledge of the process and measurement noises.**

I. Introduction

IT is well known that the Kalman Filter is equivalent to least-squares estimation when the system is linear and the process and measurement noises are mutually independent Gaussian noises [1]. Analytic or recursive solutions are available for the least-square estimation or Kalman Filter when constraints are not considered. In applications, limits or constraints on state variables are often known a priori and can be expressed as inequalities. For example temperature, pressure, flow rates, must be non-negative and cannot go above some upper bound. However, with inequalities constraints, the estimation of state becomes a constrained optimisation problem where no analytical solution may exist. With increasing time horizon, the constrained optimisation problem becomes computationally intractable, thus the idea of Moving Horizon Estimation (MHE) is used to limit the number of decision variables. MHE is a kind of optimization-based state estimation in which an estimation cost function is optimized and the measurements in a moving window are used to estimate the state. The estimation cost function often includes the error between true and estimation of the state, the error between measurement and predicted output and the arrival cost which summaries the past measurements outside of the moving window.

The relationship between full information and receding horizon estimation is investigated in [2]. A moving horizon-based approach for least-squares estimation was proposed to solve the issue of computational intractable when the constraints are taken into account [3]. The duality between control and estimation is well known in the context of Linear Quadratic Regulator and the Kalman Filter. For constrained control and estimation their Lagrangian duality is investigated in [4]. Constrained MHE for linear system and the conditions of nominal asymptotically stability are investigated in [5]. General constrained MHE for nonlinear system is proposed in [6].

This work was supported by a grant from Asian Office of Aerospace Research and Development (Ref:AOARF-124004). Dexiang Zhou, K.V. LING and E.K. Poh are with Division of Control and Instrument, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798, Singapore. (e-mail: ZHOU0180@e.ntu.edu.sg; EKVLING@ntu.edu.sg; eekpoh@ntu.edu.sg)

Constrained MHE uses the constraints (often in the form of inequalities of states and noise) to prevent the estimator from producing values which are not reasonable (i.e. the values which violate the constraints). Thus good estimation performance is often obtained using constrained MHE. In this paper constrained MHE for linear and nonlinear systems state estimation is investigated. The purpose of the paper is to demonstrate that the constrained MHE is more robust than Kalman Filter and EKF when uncertainty exists, in particular, when we have poor prior knowledge about the process and measurement noises.

This paper is organized as follows. In Section 2 we introduce the constrained MHE which is suitable for both linear and non-linear systems. In Section 3 approximations of arrival costs of constrained MHE for linear and nonlinear systems are introduced. In Section 4 the advantage of constrained MHE is shown comparing to Kalman Filter and EKF using simulations. Section 5 concludes this paper.

II. Constrained MHE

Consider the following system

$$\begin{aligned} x_{k+1} &= f\left[k, x_k\right] + \omega_k \\ y_k &= h\left[k, x_k\right] + e_k \end{aligned} \tag{1}$$

The process noise ω_k and measurement noise e_k are assumed additive, zero mean and white noises. The initial state, with estimate $\bar{x}_0$, an approximation mean, and the associated covariance matrix P_0 which is positive definite, is assumed to be uncorrelated with the two noise sequences.

We assume the system (1) are subject to the following constraints:

$$x_k \in \Omega_x,\ \omega_k \in \Omega_\omega,\ e_k \in \Omega_e \tag{2}$$

where Ω_x, Ω_ω and Ω_e are polytopic sets containing the origin.

We formulate a constrained state estimation problem as the follows:

$$\begin{aligned} J_T:\ \Phi_T^* &= \min_{\hat{x}_0,\{\hat{\omega}_k\}_{k=0}^T} \Phi_T \\ & s.t.\ \hat{x}_{k+1} = f[k,\hat{x}_k] + \hat{\omega}_k, \\ & \quad y_k = h[k,\hat{x}_k] + \hat{e}_k, \\ & \quad \hat{x}_k \in \Omega_x,\ \hat{\omega}_k \in \Omega_\omega,\ \hat{e}_k \in \Omega_e. \end{aligned} \tag{3}$$

where

$$\begin{aligned} \Phi_T =& \sum_{k=0}^{T} \left(\hat{e}_k' R^{-1} \hat{e}_k + \hat{\omega}_k' Q^{-1} \hat{\omega}_k \right) \\ &+ (\hat{x}_0 - \bar{x}_0)' P_0^{-1} (\hat{x}_0 - \bar{x}_0) \end{aligned} \tag{4}$$

and $\{\hat{\omega}_k\}_{k=0}^T$ denotes the sequence $\{\hat{\omega}_0,\ \hat{\omega}_1,\ \cdots,\ \hat{\omega}_T\}$, Q and R are the design choices of variances of the process and measurement noises. The solution of J_T at time T is $\hat{x}_{0|T}^o$ and $\{\hat{\omega}_{k|T}^o\}_{k=0}^T$. The current estimation $\hat{x}_{T|T}^o$ is the state evolution of the system (1) driven by the $\{\hat{\omega}_{k|T}^o\}_{k=0}^{T-1}$ from the initial state $\hat{x}_{0|T}^o$.

The above estimation is full information estimation without dropping old measurements. This optimization problem becomes computational intractable as T increases. MHE removes this difficulty by considering only the most recent N measurements $\{y_k\}_{T-N+1}^T$ to estimate states $\{x_k\}_{T-N+1}^T$. We reformulate the object function (4) by breaking the time interval into two pieces:

$$\begin{aligned} \Phi_T =& \sum_{k=T-N+1}^{T} \left(\hat{e}_k' R^{-1} \hat{e}_k + \hat{\omega}_k' Q^{-1} \hat{\omega}_k \right) \\ & \sum_{k=0}^{T-N} \left(\hat{e}_k' R^{-1} \hat{e}_k + \hat{\omega}_k' Q^{-1} \hat{\omega}_k \right) \\ &+ (\hat{x}_0 - \bar{x}_0)' P_0^{-1} (\hat{x}_0 - \bar{x}_0) \end{aligned} \tag{5}$$

Because we use the state space model of the system the quality of the first term of (5) only depends on $\hat{x}_{T-N+1}$ and sequences $\{\hat{\omega}_k\}_{k=T-N+1}^T$, $\{\hat{e}_k\}_{k=T-N+1}^T$. By using forward dynamic programming the full information estimation (3) with the cost function (5) can be reformulated as the following equivalence with moving horizon window:

$$\begin{aligned} J_T:\ \Phi_T^* =& \min_{\hat{x}_{T-N+1},\{\hat{\omega}_k\}_{k=T-N+1}^T} \sum_{k=T-N+1}^{T} (\hat{e}_k' R^{-1} \hat{e}_k \\ &+ \hat{\omega}_k' Q^{-1} \hat{\omega}_k) + \theta_{T-N+1}(\hat{x}_{T-N+1}) \\ & s.t.\ \hat{x}_{k+1} = f[k,\hat{x}_k] + \hat{\omega}_k, \\ & \quad y_k = h[k,\hat{\omega}_k] + \hat{e}_k, \\ & \quad \hat{x}_k \in \Omega_x,\ \hat{\omega}_k \in \Omega_\omega, \hat{e}_k \in \Omega_e \end{aligned} \tag{6}$$

where

$$\begin{aligned} \theta_T(z) = \min_{\hat{x}_0,\{\hat{\omega}_k\}_{k=0}^{T-1}} \{ & \Phi_{T-1} : \\ & \hat{x}(T;\hat{x}_0,\{\hat{\omega}_k\}_{k=0}^{T-1}) = z \} \end{aligned} \tag{7}$$

where the minimization is subject to the constraints (2) and $\hat{x}(T;\hat{x}_0,\{\hat{\omega}_k\}_{k=0}^{T-1})$ denotes the state evolution of the system (1) when the initial sate is $\hat{x}_0$ and the process noise is $\{\hat{\omega}_k\}_{k=0}^{T-1}$. It follows that $\theta_0(\hat{x}_0) = (\hat{x}_0 - \bar{x}_0)' P_0^{-1} (\hat{x}_0 - \bar{x}_0)$.

The arrival cost $\theta_{T-N+1}(\hat{x}_{T-N+1})$ denotes the contribution of the measurements $\{y_k\}_{k=0}^{k=T-N}$ on the estimation of state x_{T-N+1} and allows us to formulate the full information estimation as MHE. For the majority of systems with constraints, we can not obtain analytical expression for the arrival cost. Because full information estimation has very good theoretical properties in terms of stability and optimality [7] we formulate MHE as close as possible to the full information estimation by designing a suitable arrival cost.

III. Approximation of the Arrival Cost

One reasonable solution is to approximate the arrival cost for constrained estimation problem with the arrival cost for the unconstrained estimation problem. Here two approximations of the arrival cost for linear system and nonlinear system are introduced respectively.

For linear system it is assumed that the functions $f[k,x_k]$ and $h[k,x_k]$ are defined by

$$f[k,x_k] := A_k x_k,\ h[k,x_k] := C_k x_k.$$

The unconstrained full information estimation is known as Kalman Filter. Kalman filtering covariance update formula is

$$\begin{aligned} P_k =& Q + A_{k-1} P_{k-1} A_{k-1}' \\ &- A_{k-1} P_{k-1} C_k' (R + C_k P_{k-1} C_k')^{-1} C_k P_{k-1} A_{k-1}' \end{aligned} \tag{8}$$

with initial condition P_0 and its arrival cost is expressed as

$$\theta_k(z) = (z - \hat{x}_{k|k-1}^o)' P_k^{-1} (z - \hat{x}_{k|k-1}^o) + \Phi_{k-1}^* \tag{9}$$

In constrained MHE for linear system the arrival cost is approximated by the arrival cost of Kalman Filter (i.e. (9)).

For nonlinear system one strategy to approximate the arrival cost $\theta_k(\cdot)$ is to employ the first-order Taylor series approximation of the system (1) around the estimated trajectory. This strategy yields the Extended Kalman Filter covariance update formula. Suppose the model functions $f[k,x_k]$ and $h[k,x_k]$ are sufficiently smooth. Then the arrival cost is approximated as

$$\tilde{\theta}_k(z) = (z - \hat{x}_{k|k-1}^{MHE})' P_k^{-1} (z - \hat{x}_{k|k-1}^{MHE}) + \tilde{\Phi}_{k-1}^* \tag{10}$$

where P_j sequence is obtained by solving the matrix Riccati equation (8) in which

$$A_k = \frac{\partial f(k)}{\partial x}|_{x=\hat{x}_{k|k}^{MHE}},\ C_k = \frac{\partial h(k)}{\partial x}|_{x=\hat{x}_{k|k}^{MHE}}$$

and the superscript MHE means that the state estimation is obtained by MHE .

For $T > N$ we formulate the implementation of constrained MHE as the solution to the following least-squares program which approximates to the estimation problem J_T:

$$\begin{aligned} \tilde{J}_T:\ \tilde{\Phi}_T^* &= \min_{\hat{x}_{T-N+1},\{\hat{\omega}_k\}_{k=T-N+1}^T} \tilde{\Phi}_T \\ & s.t.\ \hat{x}_{k+1} = f[k,\hat{x}_k] + \hat{\omega}_k, \\ & \quad y_k = h[k,\hat{x}_k] + \hat{e}_k, \\ & \quad \hat{x}_k \in \Omega_x,\ \hat{\omega}_k \in \Omega_\omega,\ \hat{e}_k \in \Omega_e \end{aligned} \tag{11}$$

where the objective function is defined as

$$\tilde{\Phi}_T = \sum_{k=T-N+1}^{T} \left(\hat{e}_k^{'} R^{-1} \hat{e}_k + \hat{\omega}_k^{'} Q^{-1} \hat{\omega}_k \right) + \tilde{\theta}_{T-N+1}(\hat{x}_{T-N+1}) \quad (12)$$

Note that the arrival cost is approximated as (10) including the linear and nonlinear cases.

IV. Illustrate Example

In this section we show, with simulation examples, the advantage of constrained MHE comparing to Kalman Filter and EKF when the process and measurement noise variances are not chosen correctly. In the following simulations it is assumed that process and measurement noises are white Gaussian noises. And we denote the ith element of a vector z in time instant k as $z_{i,k}$.

A. Performance Comparison of Kalman Filter and Constrained MHE

Consider an observable discrete linear system:

$$x_{k+1} = \begin{bmatrix} 0.2695 & 0.4779 & 0.1127 & 0.1339 \\ 0.2688 & 0.4786 & 0.0753 & 0.1713 \\ 0.0053 & 0.0063 & 0.5713 & 0.0168 \\ 0.0035 & 0.0081 & 0.0094 & 0.5782 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} u_k + \omega_k$$

$$y_k = \begin{bmatrix} 1 & 0.5 & 1 & 0.5 \end{bmatrix} x_k + e_k$$

We assume the constraints are

$$\begin{bmatrix} 2.2 \\ 2.4 \\ 0 \\ 0 \end{bmatrix} \leqslant x_k \leqslant \begin{bmatrix} 4.3 \\ 4.2 \\ 2.7 \\ 2.3 \end{bmatrix}, \quad \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \leqslant w_k \leqslant \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

and

$$E\left(\omega_k \omega_k^T\right) = 0.1 \times I_4, \; E\left(e_k e_k^T\right) = 5.$$

Kalman Filter results is an optimal estimation for linear system when Q and R are chosen to match process and measurement noises. However, the variances of process and measurement noises are often not known exactly in practice and Q and R are often used as tuning parameters of the Kalman filter.

Figure 1 and 2 show that the Kalman Filter could produce estimates that violate the state constraints when Q and R are not chosen correctly while the constrained MHE gives state estimates that are within the constraints. The reason is that the prior information of constraints are taken into account in the constrained MHE while Kalman Filter does not.

Also in Table I we use the criterion:

$$J_{KF} = \sum_{k=1}^{T} (x_{1,k} - \hat{x}^o_{1,k|k})^2, \; J_{MHE} = \sum_{k=1}^{T} (x_{1,k} - \hat{x}^{MHE}_{1,k|k})^2$$

to compare the estimation performance of Kalman Filter and MHE when the variances of process and measurement noises are not chosen correctly. The correct values are $q = 0.1$ and $R = 5$ (row 1). When this is the case, Kalman Filter gives the optimal estimates and the performance of Kalman Filter and constrained MHE is almost the same. The performance of Kalman Filter deteriorates significantly when the values of q and R are inaccurate (rows 2 to 4). The constrained MHE, however, is less sensitive to the values of q or R. Thus constrained MHE is robust to the uncertain information, in particular to the variances of process and measurement noises.

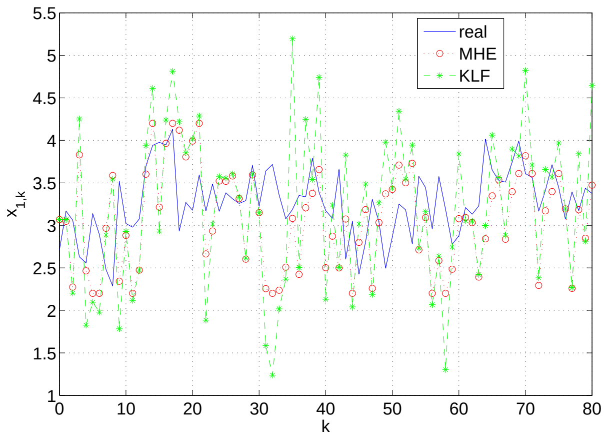


Fig. 1. Estimation performance with $Q = 10 \times I_4$ and $R = 5$

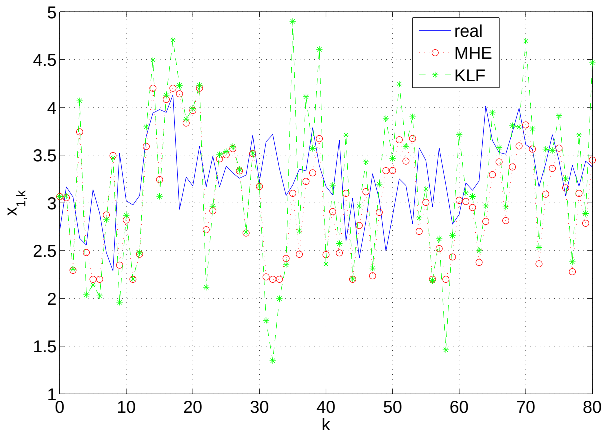


Fig. 2. Estimation performance with $Q = 0.1 \times I_4$ and $R = 0.1$

TABLE I
Performance Comparison of Kalman Filter and MHE

$Q = qI_4$ with q	R	Kalman Filter	MHE wih $N = 5$
0.1	5	13.9794	13.9794
10	5	166.6161	37.1588
10	1	194.8563	37.5834
100	1	203.5847	37.7831

B. Performance Comparison of EKF and Constrained MHE

Consider a radar tracking application

$$x_{k+1} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k + \omega_k$$

$$y_k = \begin{bmatrix} \sqrt{x_{1,k}^2 + x_{3,k}^2} \\ \arctan \frac{x_{3,k}}{x_{1,k}} \end{bmatrix} + e_k$$

where ω_k and e_k are zero means Gaussian noise with variance

$$E\left(\omega_k \omega_k^T\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}, \; E\left(e_k e_k^T\right) = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}.$$

The constraints are

$$140 \leqslant x_{2,k}, \; x_{4,k} \leqslant 210.$$

The initial state is $x_0 = [0 \; 200 \; 0 \; 200]'$ and we chose

$$P_0 = I_4, \; N = 40.$$

Figure 3 shows that the performance of EKF and constrained MHE are almost the same when Q and R are chosen correctly. Constrained MHE is able to keep the state estimates within the constraints while the EKF is unable to do so shown in Figure 4 when the values of Q and R are not chosen correctly. The inaccurate values of Q and R used in Figure 4 are

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R = \begin{bmatrix} 0.09 & 0 \\ 0 & 0.1 \end{bmatrix}$$

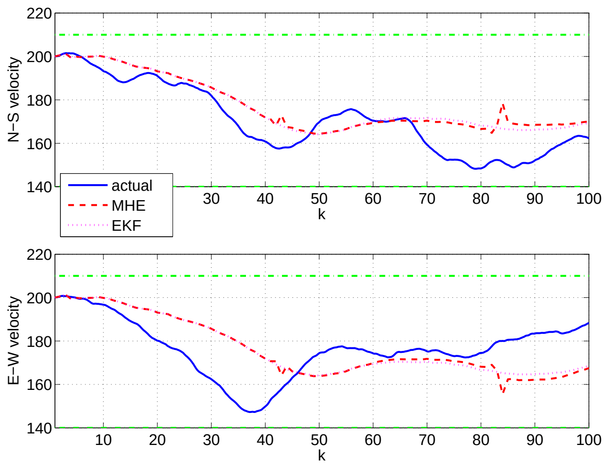


Fig. 3. Estimation of Velocities using EKF and constrained MHE when Q and R are chosen correctly

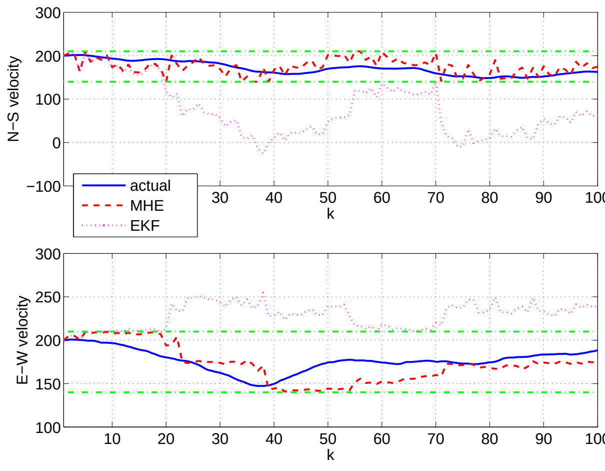


Fig. 4. Estimation of Velocities using EKF and constrained MHE when Q and R are inaccurate

V. Conclusion

In this paper, we review the implementation of constrained MHE which is applicable to both linear and nonlinear systems. The constrained MHE is formulated where the arrival cost is approximated by the solution to a Kalman Filter for a linear estimation problem and by EKF for nonlinear estimation. Two illustrative examples, including one on a typical non-linear radar tracking problem, were studied and the simulation results demonstrate that the constrained MHE exhibits robust performance when there is poor prior knowledge of the process and measurement noises as compared to stand-alone Kalman Filter and EKF. Our future work will consider deploying the proposed MHE strategy for real-time applications where computational resources are limited. We aim to devise algorithms that could exploit the structure of the specific problems and types of nonlinearity to gain computational advantage.

References

[1] H.W. Sorenson. Least-squares estimation: from gauss to kalman. *Spectrum, IEEE*, 7(7):63–68, 1970.
[2] KV Ling and KW Lim. Receding horizon recursive state estimation. *Automatic Control, IEEE Transactions on*, 44(9):1750–1753, 1999.
[3] D.G. Robertson, J.H. Lee, and J.B. Rawlings. A moving horizon-based approach for least-squares estimation. *AIChE Journal*, 42(8):2209–2224, 2004.
[4] G.C. Goodwin, J.A. De Doná, M.M. Seron, and X.W. Zhuo. Lagrangian duality between constrained estimation and control. *Automatica*, 41(6):935–944, 2005.
[5] C.V. Rao, J.B. Rawlings, and J.H. Lee. Constrained linear state estimationa moving horizon approach. *Automatica*, 37(10):1619–1628, 2001.
[6] C.V. Rao, J.B. Rawlings, and D.Q. Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *Automatic Control, IEEE Transactions on*, 48(2):246–258, 2003.
[7] B.D.O. Anderson and J.B. Moore. *Optimal filtering*, volume 11. Prentice-hall Englewood Cliffs, NJ, 1979.

Moving Horizon Estimation on a Chip

Dang Van Thuy
Interdisciplinary Graduate School
Nanyang Technological University
Singapore 639798
Email: DANG0028@e.ntu.edu.sg

K.V. LING
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore 639798
Email: EKVLING@ntu.edu.sg

***Abstract*—Second order Quadratic Programming (QP) solvers such as interior-point method (IPM) require the solution of a system of linear equations at every iteration and could be a factor limiting the implementation of IPM to miniaturized devices or embedded systems. In contrast, first order QP solvers such as alternating direction method of multipliers (ADMM) does not require the solution of a system of linear equations. Thus first order QP solver is cheaper and easier to be implemented in embedded systems such as FPGA which has limited hardware resources. In this paper an FPGA implementation of ADMM which solves QP problems arising from Moving Horizon Estimation is proposed to demonstrate the "MHE on a Chip" idea. Our design has been implemented in both fixed-point and floating point arithmetic on the Xilinx Zynq-7000 XC7Z020-1CLG484C AP SoC and clocks at 50 MHz.**

I. Introduction

Interior Point Method (IPM) and Active Set Method (ASM) are two commonly employed approaches for solving QP problems. In every iteration of IPM and ASM, the solution of a system of linear equations is required and this could be a factor limiting the implementation IPM or ASM to miniaturized devices or embedded systems. Recently first order QP solvers receive significant interests [1]–[3]because of its simple computational structure and low cost implementation. In contrast to the IPM and ASM, first order QP solver such as alternating direction method of multipliers (ADMM) does not require the solution of a system of linear equations and this makes first order QP solver easier and cheaper to implement on embedded systems such as Field Programmable Gate Array (FPGA).

Recent advances in reconfigurable hardware technology have made FPGA becoming more popular in a wide range of embedded applications. Implementation with FPGA brings a shorter design cycle and greater flexibility comparing with ASIC. Moreover, high speed computation can be achieved through parallelization and customization to meet timing requirement.

There have been several previous FPGA implementations of QP solvers [4]–[6]. In [6] QP solver based on infeasible IPM has been implemented on FPGA for Model Predictive Control (MPC). The comparison of implementation of IPM and ASM on FPGA for MPC was discussed in [7]. In [8] the authors proposed that the FPGA can provide substantial accelerating by exploiting the parallelism inherent in Multiplexed MPC (MMPC). All these papers implemented the QP solvers which requires the solution of system of linear equations. Similar to Fast Gradient Method, ADMM is also a first order method that has simple computational structures which allow efficient parallelism. In addition, the method are division-free making the implementation efficiently since division is a costly operation in FPGA design, in term of resource and latency, comparing with other operations such as multiplication or addition .

Using high-level synthesis (HLS) tools, such as Vivado HLS from Xilinx or Synphony C Compiler from Synopsys [9], for implementation with FPGA brings lots of advantages. Designing at a higher abstraction level makes it possible to handle the increasing complexity of embedded applications. For those who do not have the training to work with hardware description languages (HDLs) to perform a register transfer level (RTL) description of the hardware, HLS tools offer an alternative approach. HLS tools also facilitate rapid prototyping. FPGA design and implementation by HLS not only significantly saves time, but also reduces the risk of mistakes. For the implementation of ADMM-based QP Solver on FPGA, we used Vivado HLS and System Generator to generate the bit stream from a C-based design.

MHE is a kind of optimization-based state estimation where an on-line constrained optimization is solved and the measurements in a moving window are used to estimate the state. In applications, limits or constraints on state variables are often known a priori and can be expressed as inequalities. For example temperature, pressure, flow rates, and concentration, must be non-negative and cannot go above some upper bound. Constrained MHE can include the prior information (i.e. constraints), non-Gaussian noises [10] and nonlinear system without linearisation [11]. The parallel computing architecture of FPGA can accelerate the computation to solve on-line constrained optimization. In this paper a ADMM-based QP solver which solves on-line constrained optimization problems is implemented in FPGA with the aim to extend MHE to embedded applications.

This paper is organized as follows. In Section II we summarise the method of using ADMM to solve MHE for linear system. The procedure for designing and arriving at a FPGA implementation of ADMM-based MHE is described in Section III. Section IV discusses the approaches to accelerate computation for our FPGA designs. In Section V, an example is used to illustrate our designs. Conclusions and future work are given in Section VI.

II. ADMM METHOD FOR MHE

A. QP formulation of MHE

Consider the following linear system

$$x_{k+1} = Ax_k + Bu_k + \omega_k \tag{1}$$
$$y_k = Cx_k + e_k \tag{2}$$

where $x_k \in \mathbb{R}^n$ is the state vector, $y_k \in \mathbb{R}^p$ is the measurement. The process noise ω_k and measurement noise e_k are assumed additive, zero mean and white noises with variances Q and R respectively. The initial state, with estimate $\bar{x}_0$, an approximation mean, and the associated covariance matrix P_0 which is positive definite, is assumed to be uncorrelated with the two noise sequences.

We assume that the state and noise vectors in (1) are subject to the following constraints:

$$x_k \in \Omega_x,\ \omega_k \in \Omega_\omega \tag{3}$$

where Ω_x and Ω_ω are polytopic sets containing the origin.

Given the N most recent measurements $\{y_k\}_{k=T-N+1}^{T}$ at T, the MHE is defined as

$$\begin{aligned}
\min_{\substack{\hat{x}_{T-N+1},\\ \{\hat{\omega}_k\}_{k=T-N+1}^{T-1}}} \ & \frac{1}{2}\sum_{k=T-N+1}^{T} \|\hat{e}_k\|_{R^{-1}}^2 + \frac{1}{2}\sum_{k=T-N+1}^{T-1} \|\hat{\omega}_k\|_{Q^{-1}}^2 \\
& + \frac{1}{2}\|\hat{x}_{T-N+1} - \hat{\mu}_{T-N+1}\|^2_{P^{-1}_{T-N+1}} \\
\text{s.t. } & \hat{x}_{k+1} = A\hat{x}_k + Bu_k + \hat{\omega}_k, \\
& y_k = C\hat{x}_k + \hat{e}_k, \\
& \hat{x}_k \in \Omega_x,\ \hat{\omega}_k \in \Omega_\omega
\end{aligned} \tag{4}$$

where

$$\begin{aligned}
P_k =& Q + AP_{k-1}A' \\
& - AP_{k-1}C'(R + CP_{k-1}C')^{-1}CP_{k-1}A'
\end{aligned} \tag{5}$$
$$\hat{\mu}_{T-N+1} = A\hat{x}^o_{T-N} + Bu_{T-N} \tag{6}$$

and $\hat{x}^o_{T-N}$ is the estimation of x at $T-N$.
By defining:

$$z = \begin{bmatrix}\hat{x}^T_{T-N+1} & \cdots & \hat{x}^T_T & \hat{\omega}^T_{T-N+1} & \cdots & \hat{\omega}^T_{T-1}\end{bmatrix}^T$$

$$Y = \left[\hat{\mu}^T_{T-N+1}, y^T_{T-N+1}, \cdots, y^T_T\right]^T.$$

$$H = \begin{bmatrix}\Upsilon^T \Lambda \Upsilon & 0 \\ 0 & I_{N-1}\otimes Q^{-1}\end{bmatrix},$$

$$\mathbb{K} = \begin{bmatrix} 1_N \otimes \Omega_x \\ 1_{N-1}\otimes \Omega_\omega\end{bmatrix} \tag{7}$$

$$g = -\left(I_{N-1}\otimes B\right)\begin{bmatrix}u_{T-N+1}\\ \vdots \\ u_{T-1}\end{bmatrix},\ h = \begin{bmatrix}\Phi Y \\ 0\end{bmatrix}, \tag{8}$$

$$G = \begin{bmatrix}
A & -I_n & 0 & \cdots & 0 & 0 & I_n & 0 & \cdots & 0 \\
0 & A & -I_n & \cdots & 0 & 0 & 0 & I_n & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & A & -I_n & 0 & 0 & \cdots & I_n
\end{bmatrix}$$

and

$$\Phi = -\Upsilon^T \Lambda, \tag{9}$$

$$\Upsilon = \begin{bmatrix} I_n \\ I_N \otimes C\end{bmatrix},\ \Lambda = \begin{bmatrix} P^{-1}_{T-N+1} & 0 \\ 0 & I_N \otimes R^{-1}\end{bmatrix},$$

($\otimes$ is Kronecker product, I_m is denoted as the identity matrix with m of dimension,)

The constrained optimization problem (4) can be formulated as following standard form:

$$\min_z \ \frac{1}{2}z^T Hz + h^T z \tag{10}$$
$$\text{s.t. } Gz = g, \tag{11}$$
$$z \in \mathbb{K}, \text{(defined as Eq. (7))} \tag{12}$$

The size of H is $(2N-1)n \times (2N-1)n$ and QP size is defined as

$$QPSize = (2N-1)n \tag{13}$$

B. ADMM for MHE

The ADMM algorithm [12] for the MHE QP problem (10) is as follow [13]:

```
– Choose initial condition $z_0$, $\theta_0, \tau_0$
Do
  1. $\theta_{i+1} = M_{11}(-h - \tau_i + \rho z_i) + M_{12}g$        (14)
  2. $z_{i+1} = \pi_{\mathbb{K}}(\theta_{i+1} + \frac{1}{\rho}\tau_i)$        (15)
  3. $\tau_{i+1} = \rho\theta_{i+1} + \tau_i - \rho z_{i+1}$        (16)
  4. $\varepsilon = \|\theta_{i+1} - \theta_i\|^2$
While($\varepsilon > \varepsilon_0$)
                                                        (17)
```

where M_{11} and M_{12} are computed offline as

$$\begin{bmatrix} H + \rho I_{(2N-1)n} & G^T \\ G & 0\end{bmatrix}^{-1} = \begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22}\end{bmatrix} \tag{18}$$

with the notation of projection operation:

$$\pi_{\mathbb{K}}(z_k) = \arg\min_{z\in\mathbb{K}} \|z - z_k\|^2 \tag{19}$$

In MHE application constraints are usually imposed to restrict the minimum and maximum values of internal states or noises. For simplicity of FPGA implementation we assume the constraints Ω_x and Ω_ω are box type constraints, i.e.,

$$\Omega_x = \{x_k : x_{\min} \leqslant x_k \leqslant x_{\max}\} \tag{20}$$
$$\Omega_\omega = \{\omega_k : \omega_{\min} \leqslant \omega_k \leqslant \omega_{\max}\} \tag{21}$$

Consequence, $\mathbb{K}$ is box: $\{\mathbb{K} : z_{min} \leq z \leq z_{max}\}$.
where

$$z_{\min} = \begin{bmatrix} 1_N \otimes x_{\min} \\ 1_{N-1} \otimes \omega_{\min} \end{bmatrix}, \; z_{\max} = \begin{bmatrix} 1_N \otimes x_{\max} \\ 1_{N-1} \otimes \omega_{\max} \end{bmatrix}$$

and 1_m is the $m \times 1$ vector of ones. With the box type constraints assumption the solution (15) reduces to

$$z_{i+1} = \max\left\{z_{\min},\; \min\left\{z_{\max},\; \left(\theta_{i+1} + \frac{1}{\rho}\tau_i\right)\right\}\right\} \tag{22}$$

As a summary, MHE with box constraints using ADMM algorithm is:

Algorithm 1

```
– Compute  $\mathtt{M}_{11}, \mathtt{M}_{12}$ with a $\rho > 0$ as (18)
– Choose initial condition $\mathtt{z}_0, \theta_0 \in \mathbb{K}$,
   $\tau_0 = 0, \varepsilon_0$ : Stopping Criterion
– **While** (new measurement is available)
   Compute h and g as (8), $\mathtt{M}_{12}\mathtt{g}$
   **Do**
     1. $\theta_{i+1} = \mathtt{M}_{11}(-\mathtt{h} - \tau_i + \rho \mathtt{z}_i) + \mathtt{M}_{12}\mathtt{g}$        (23)
     2. $\mathtt{z}_{i+1} =$
        $\max\left\{\mathtt{z}_{\min}, \min\left\{\mathtt{z}_{\max}, \left(\theta_{i+1} + \frac{1}{\rho}\tau_i\right)\right\}\right\}$        (24)
     3. $\tau_{i+1} = \rho\theta_{i+1} + \tau_i - \rho \mathtt{z}_{i+1}$        (25)
     4. $\varepsilon = \|\theta_{i+1} - \theta_i\|^2$        (26)
   **while**$(\varepsilon > \varepsilon_0)$
– **end While**
```

III. FPGA Implementation

To implement a ADMM-based QP solver for MHE on FPGA, we use Vivado HLS, Xilinx System Generator and MATLAB/Simulink as development tools, and the Zynq ZC702 board [14]–[16].

Vivado HLS, a high-level synthesis tool, allows the designer to describe the computational algorithm using a high-level language, commonly C/C++/SystemC, to generate automatically the register transfer level (RTL) description, e.g. VHDL or Verilog, for FPGA implementation. Especially for those whose main expertise is in control system design, writing algorithms in C is easier and more familiar than using hardware description languages. Therefore, high-level synthesis tools accelerate the design process for complicated algorithms, and significantly reduce design time. The main steps to implement our ADMM-based QP sovler on FPGA for MHE problems is shown in Figure 1 and is described next.

Step 1: C/C++ Coding
The MHE with box constraints using ADMM algorithm (Algorithm 1) is written in C where the ADMM part is shown in Figure 2.

Step 2: Synthesis and RTL export with Vivado HLS

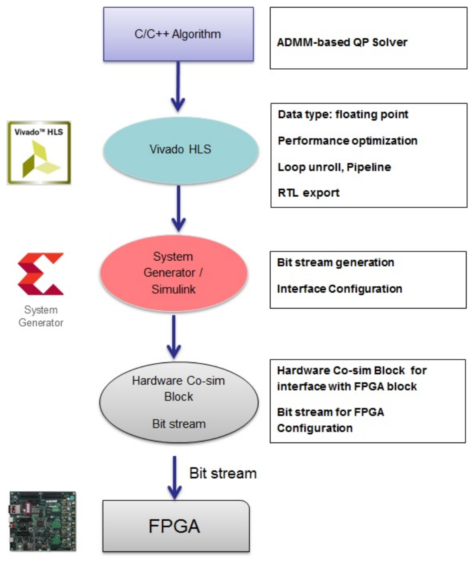
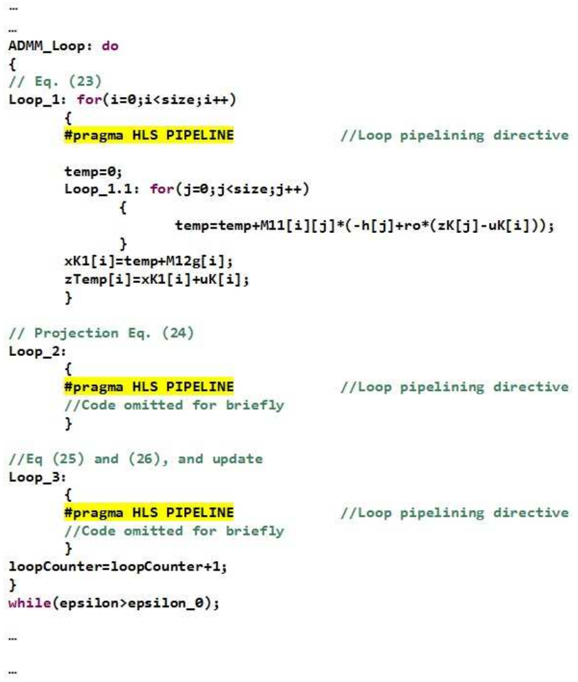


Fig. 1. Design flow for FPGA Implementation



```
…
…
ADMM_Loop: do
{
// Eq. (23)
Loop_1: for(i=0;i<size;i++)
        {
        #pragma HLS PIPELINE               //Loop pipelining directive

        temp=0;
        Loop_1.1: for(j=0;j<size;j++)
                {
                        temp=temp+M11[i][j]*(-h[j]+ro*(zK[j]-uK[i]));
                }
        xK1[i]=temp+M12g[i];
        zTemp[i]=xK1[i]+uK[i];
        }

// Projection Eq. (24)
Loop_2:
        {
        #pragma HLS PIPELINE               //Loop pipelining directive
        //Code omitted for briefly
        }

//Eq (25) and (26), and update
Loop_3:
        {
        #pragma HLS PIPELINE               //Loop pipelining directive
        //Code omitted for briefly
        }
loopCounter=loopCounter+1;
}
while(epsilon>epsilon_0);

…

…
```

Fig. 2. ADMM C code in the MHE with box constraints

Vivado HLS supports floating-point (single or double precision IEEE-754 standard compliance) or fixed-point arithmetic for synthesis. In addition, Vivado HLS allows designers to customize the design by adding synthesis directives/constraint such as loop-unroll, pipeline, resource constraint, etc for performance optimization. The design is then synthesized and exported as a RTL module with different options for further processing with other Xilinx tools.

Step 3: Bit stream and Hardware co-simulation block generation with System Generator
To verify that the ADMM algorithm is implemented correctly, the RTL module generated from Vivado HLS is imported to System Generator (SysGen) which is integrated with Simulink. SysGen generates the bit-stream for FPGA configuration, and a Hardware co-sim block for the interfacing between FPGA and Simulink environment [14].

Step 4: Implementation on FPGA
Through the hardware co-sim block generated from SysGen, the bit stream is downloaded to the FPGA board. This block allows Simulink and the FPGA board to be connected directly so that FPGA-in-the-Loop simulation can be carried out as shown in Figure 3. In this simulation scheme, QP problems of MHE at each sample time are sent to FPGA board via JTAG communication from Simulink. After completing the computation, FPGA board sent back the result to Simulink, and wait for the next computation task. The interface setting between the FPGA board and Simulink is defined by the Hardware Co-sim block. In the simulation, the FPGA clocking mode is free running. The synchronization mechanism (Figure 4) is as follow : the Logical Control block generates the Start signal periodically (Ts), this signal triggers the FPGA starting operation; when FPGA is busy the Idle signal goes low, and goes high when the FPGA has completed the computation. Based on the rising edge of the Idle signal, the Buffer capture the solution.

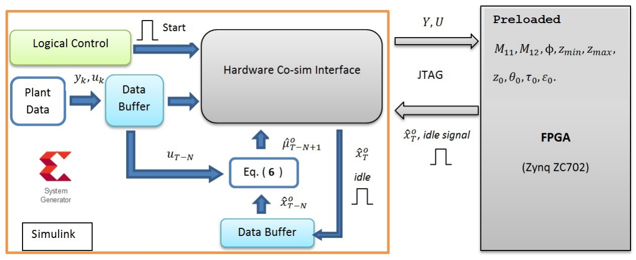


Fig. 3. Hardware Co-simulation with FPGA

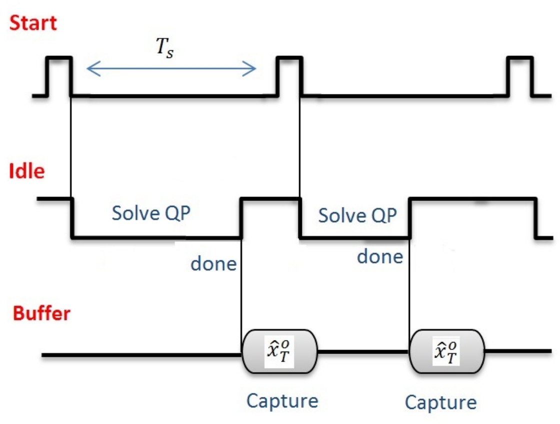


Fig. 4. Synchronization mechanism

IV. Computational Acceleration

Since our proposed ADMM for MHE (Algorithm 1) is division free, it allows an efficient fixed-point arithmetic implementation which will run faster than a floating point design. Moreover, ADMM consist loops involving in computing matrix-vector product, loop pipeline technique was employ to exploit the parallelism. The efficiency of fixed-point arithmetic and loop pipelining are discussed in Section V.

A. *Fixed-point Arithmetic Implementation*

Comparing with floating point arithmetic, the circuitry for fixed-point computations is much more simple and faster, but the dynamic range is limited. The design of fixed point data type is based on the accuracy required, and the dynamic range of the variable involving in the algorithm. Since there have not been any general results on establishing an analytical bound for ADMM, it must estimate the bounds through simulation study for a particular QP problem. In our particular MHE example shown in Section V, through simulation we got the lower and upper bound for dynamic variables (Table 1). The fixed point data type is chosen (Table I) to ensure the required accuracy and dynamic range. For an accuracy of $\epsilon = 10^{-6}$, about $(6ln(10))/ln(2) = 20$ fractional bits is required. Looking at the different range of the variables, ideally different word and fractional bit length fixed-point representation should be used. Since the DSP48 block of Xilinx Zynq ZC702 consists of a 25×18 bit multiplier, we chose 25 and 18 bits long data types for variables involving in the multiplication operation. The use of different fixed-point data type is a subject of the future work. In order to show the effectiveness of fixed point arithmetic, we also implemented the algorithm in floating point (single precision 32 bit IEEE 754 standard) for resouces and timing comparison (Section V).

TABLE I
Fixed point designation ($N = 5, n = 4$)

Variable	Range	Word	Fraction
M_{11},M_{12}	$(-0.02, 0.09)$	25	19
z	$(-0.12, 4.3)$	25	19
θ	$(-0.12, 4.56)$	25	19
τ	$(0, 1.31)$	25	19
h	$(-15.02, 0)$	25	19
g	$(-1, 1)$	25	19
$ro, 1/ro$	$ro = 4, 1/ro = 0.25$	18	10
$Intermediate$	$(-10, 10)$	25	19

B. *Loop pipelining*

To exploit the parallelism, loop pipelining (LPP) technique was employed. The computation is in a parallel manner. As a result, this reduces the latency, but the cost is more hardware resource and hence chip area. To apply LPP, Vivado HLS provides compiler directive: `set_directive_pipeline <code section>` or `# pragma HLS pipeline [Option]` [14]. In default option, Vivado HLS automatically control the *Initiation Interval* (II) which is the number of cycles between each new execution of the loop body, to achieve minimum latency. Figure 5 illustrates LPP mechanism. If a loop need Q iterations, each iteration takes N cycles to complete, then it take NQ cycles to complete (Figure 5a). When employ LPP, the latency of the loop is $(Q-1)II+N$ cycles (Figure 5b). The lower the II is, the smaller latency it achieves. In our design, three loops of the ADMM algorithm is applied pipeline directive, by adding *pragma directive* before the code section of the loop as in Figure 2. The accelerating effectiveness depends on the computational architecture and data dependencies of the algorithm. Since ADMM consist of mainly dot-product operations, which is simple computation structure and high data dependence, so that it allows efficient parallelism. In Section V we will show the computation acceleration and resources usage of ADMM with LPP.

In summary, Table II shows the various designs implemented on FPGA.

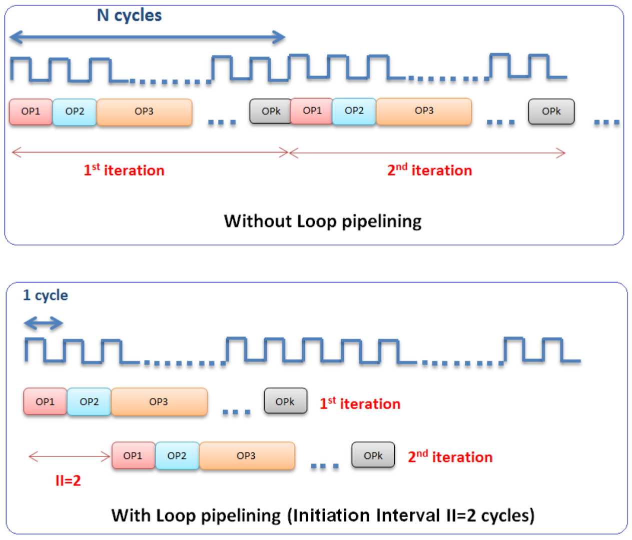


Fig. 5. Loop pipelining

TABLE II
VARIOUS "MHE ON CHIP" DESIGNS IMPLEMENTED

Design Name	Description
Fl32	Single precision floating point without LPP
Fl32-LPP	Single precision floating point with LPP
Fi25	25-bit word fixed-point without LPP
Fi25-LPP	25-bit word fixed-point with LPP

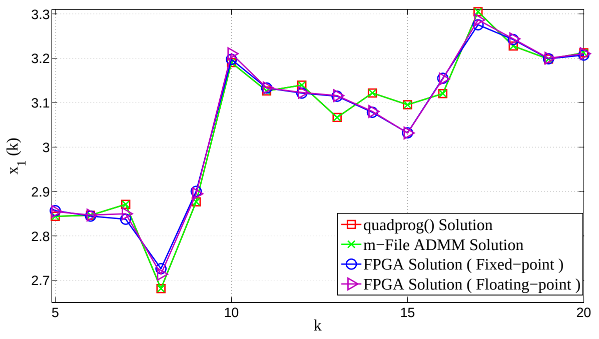


Fig. 6. Comparison of MHE solutions by FPGA and PC

V. Results

A. Illustrative Example

As an illustration, we use our designs shown in Table II to solve the following MHE example:

$$x_{k+1} = \begin{bmatrix} 0.2695 & 0.4779 & 0.1127 & 0.1339 \\ 0.2688 & 0.4786 & 0.0753 & 0.1713 \\ 0.0053 & 0.0063 & 0.5713 & 0.0168 \\ 0.0035 & 0.0081 & 0.0094 & 0.5782 \end{bmatrix} x_k + \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} u_k$$
$$+ \omega_k$$
$$y_k = \begin{bmatrix} 1 & 0.5 & 1 & 0.5 \end{bmatrix} x_k + e_k$$

We assume the constraints are

$$\begin{bmatrix} 2.2 \\ 2.4 \\ 0 \\ 0 \end{bmatrix} \leqslant x_k \leqslant \begin{bmatrix} 4.3 \\ 4.2 \\ 2.7 \\ 2.3 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \leqslant w_k \leqslant \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

and

$$Q = E\left(\omega_k \omega_k^T\right) = 0.1 \times I_4,\ R = E\left(e_k e_k^T\right) = 5,$$
$$x_0 = \begin{bmatrix} 3 & 3 & 1.3 & 1.3 \end{bmatrix}^T,\ u_k = 0.5.$$

The parameters of MHE (4) are

$$P_0 = I_4,\ \hat{\mu}_0 = \begin{bmatrix} 3 & 3 & 1.3 & 1.3 \end{bmatrix}^T,\ N = 5.$$

B. Hardware Co-simulation Result

The MHE solutions obtained from employing MATLAB quadprog(), m-File ADMM and ADMM implemented on FPGA are depicted in Figure 6 which shows that the MHE solutions in MATLAB on a PC (red-square line and green cross line) and MHE solution by ADMM method in FPGA (blue-rhombus line and pink-diamond) are slightly different. The fixed-point version is almost the same as the floating-point version on FPGA. Figure 7 show the relative errors in FPGA computation(both fixed-point and floating point), and from this we see that the accuracy are acceptable (less than 2%)

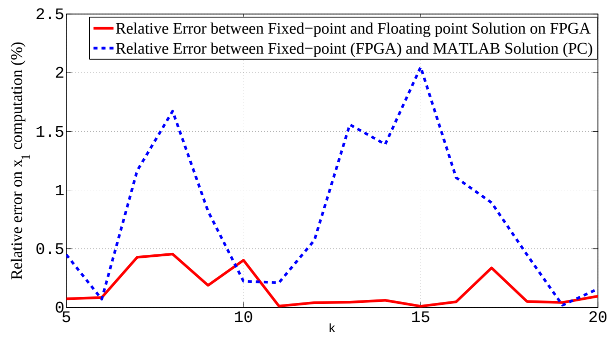


Fig. 7. Relative Error between different computations

C. Average running time and iteration time

Table III shows the average running time per MHE sampling instant of different FPGA designs for different QP sizes. The QP size depends on the estimation horizon N and is equal to $(2N-1).4$ (see (13)) in our MHE example. The second column of Table III shows the average iteration time, to achieve an accuracy $\epsilon_0 = 10^{-6}$, at each sampling instant of ADMM in FPGA . The number of iterations of ADMM increases as the QP Size grows. Moreover, the number of iterations of ADMM also depends on the numerical characteristic of a particular QP problem, e.g the condition number of the Hessian matrix.

Computational acceleration of LPP and fixed-point designs

From column 6 & 9 of Table III, we see that LPP sped up the computation significantly. For a fixed-point design, LPP speeds up about 7 times, whereas with a floating point design, it is about 30 times faster with LPP.

Also as expected, fixed-point design runs faster than floating point design. From two last column of Table III, we can see that the speed up factor is about 4.7 and 1.2 for designs with and without LPP respectively.

Resource Comparison

As expected, fixed-point design is not only faster, but also it cost less hardware resources than floating point design. Also, a design with LPP which speed up the computation costs much more resource than a design without LPP. As an illustration, Table IV shows the resource usage for the case when the estimation horizon $N = 5$ resulting in QP Size= 36.

TABLE IV
HARDWARE RESOURCE COMPARISON (QP SIZE 36)

Design	BRAM18s	DSP48s	LUTs	FFs
$Fi25$	16	14	1385	837
$Fi25 - LPP$	16	157	5377	3977
$Fl32$	17	16	1538	3521
$Fl32 - LPP$	17	32	8976	8562
Available	280	220	53200	106400

D. Max QP size

Base on the resource used for some difference QP Size problems; we estimate the largest QP size that can be implemented on this FPGA board is about 300. This is the largest solvable QP problem on an embedded platform in the literature. Future work will be carried out to confirm this number.

VI. CONCLUSION AND FUTURE WORKS

In this paper we have implemented the ADMM method on an FPGA to solve QP problems arising from MHE, and demonstrated the feasibility of the "MHE on chip" idea. The average running time per MHE sampling instance by various FPGA designs are compared for different QP sizes. Also we compare the speed and resources usage of our MHE-on-FPGA implementations with and without Loop pipelining as well as fixed-point and floating point arithmetic. Loop pipelining technique sped up significantly the computation but much more resources used. Fixed-point design not only faster but also cheaper than a floating point design. A natural extension of this work includes

1) Establish the generally analytical bound for fixed-point arithmetic implementation, since the current bound is based on the simulation
2) Accelerate computation on FPGA by further exploring the structure and property of ADMM and MHE
3) ADMM-based MHE on FPGA for nonlinear plant and apply the ADMM implemented on FPGA for MHE to a real plant.

REFERENCES

[1] Stefan Richter, Colin Neil Jones, and Manfred Morari. Real-time input-constrained mpc using fast gradient methods. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 7387–7393. IEEE, 2009.
[2] Stefan Richter, Colin Neil Jones, and Manfred Morari. Computational complexity certification for real-time mpc with input constraints based on the fast gradient method. *Automatic Control, IEEE Transactions on*, 57(6):1391–1403, 2012.
[3] Panagiotis Patrinos and Alberto Bemporad. An accelerated dual gradient-projection algorithm for linear model predictive control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 662–667. IEEE, 2012.
[4] Juan L Jerez, Paul J Goulart, Stefan Richter, George A Constantinides, Eric C Kerrigan, and Manfred Morari. Embedded predictive control on an FPGA using the fast gradient method. In *Proc. European Control Conf., Zürich, Switzerland*, 2013.
[5] Hartley E.N. and Maciejowski J.M. Graphical FPGA design for a predictive controller with application to spacecraft rendezvous. *Accepted for presentation at CDC '13*, 2013.
[6] K.V. Ling S.P. Yue and J.M. Maciejowski. A FPGA implementation of model predictive control. In *Proc. American Control Conference (ACC), 2006*, pages 1930–1935. IEEE, 2006.
[7] Mark SK Lau, SP Yue, KV Ling, and JM Maciejowski. A comparison of interior point and active set methods for FPGA implementation of model predictive control. In *Proc. European Control Conference*, pages 156–160, 2009.
[8] Juan L Jerez, George A Constantinides, Eric C Kerrigan, and Keck-Voon Ling. Parallel mpc for real-time FPGA-based implementation. In *Proceedings of the 18th IFAC World Congress*, volume 18, 2011.
[9] Synopsys Inc. *Synphony C Compiler Datasheet*, 2013.
[10] Douglas G Robertson and Jay H Lee. On the use of constraints in least squares estimation and control. *Automatica*, 38(7):1113–1123, 2002.

TABLE III
COMPARISON OF AVERAGE RUNNING TIME PER SAMPLING INSTANT

N	QP Size	Number of Iteration	Fixed-point(FPGA)			Floating-point(FPGA)			Fixed-point speed up factor[3]	
			Fi25 (ms)[1]	Fi25-LPP (ms)[1]	Speed up factor[2] $\frac{Fi25}{Fi25-LPP}$	Fl32 (ms)[1]	Fl32-LPP (ms)[1]	Speed up factor[2] $\frac{Fl32}{Fl32-LPP}$	$\frac{Fl32}{Fi25}$	$\frac{Fl32-LPP}{Fi25-LPP}$
5	36	50	5.5	0.82	6.7	25.5	1.01	25.82	4.69	1.23
6	44	67	13.7	1.6	6.9	51.18	1.91	26.9	4.61	1.18
8	60	80	24.2	3.4	7.1	112.70	3.85	29.3	4.66	1.13
10	76	90	43.2	5.9	7.3	202.25	6.62	30.6	4.68	1.12
12	92	98	68.5	9.2	7.4	321.49	11.68	27.5	4.69	1.26

[1] FPGA runs at 50 MHz.
[2] Speed up factor of LPP design
[3] Speed up factor of fixed-point design

[11] Christopher V Rao, James B Rawlings, and David Q Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *Automatic Control, IEEE Transactions on*, 48(2):246–258, 2003.

[12] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and $Trends^{®}$ in Machine Learning*, 3(1):1–122, 2011.

[13] Juan L Jerez, Paul J Goulart, Stefan Richter, George A Constantinides, Eric C Kerrigan, and Manfred Morari. Embedded online optimization for model predictive control at megahertz rates. *arXiv:1303.1090 [cs.SY]*, 2013.

[14] Xilinx. *Vivado Design Suite User Guide-High-Level Synthesis UG902 (v2012.2)*, 2012.

[15] Xilinx. *ZC702 Evaluation Board for the Zynq-7000 XC7Z020 All Programmable SoC User Guide UG850 (v1.2)*, 2013.

[16] Xilinx. *System Generator for DSP User Guide UG640 (v 14.3)*, 2012.

Appendex to Final Report for AOARD Grant 124004

Moving Horizon Estimation on a Chip

K. V. Ling (ekvling@ntu.edu.sg)
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
Phone: +65 6790 5567   Fax: +65 6793 3318

Period of Performance: 1 June 2012 - 31 May 2014

A Simulink Model for "MHE on a Chip"

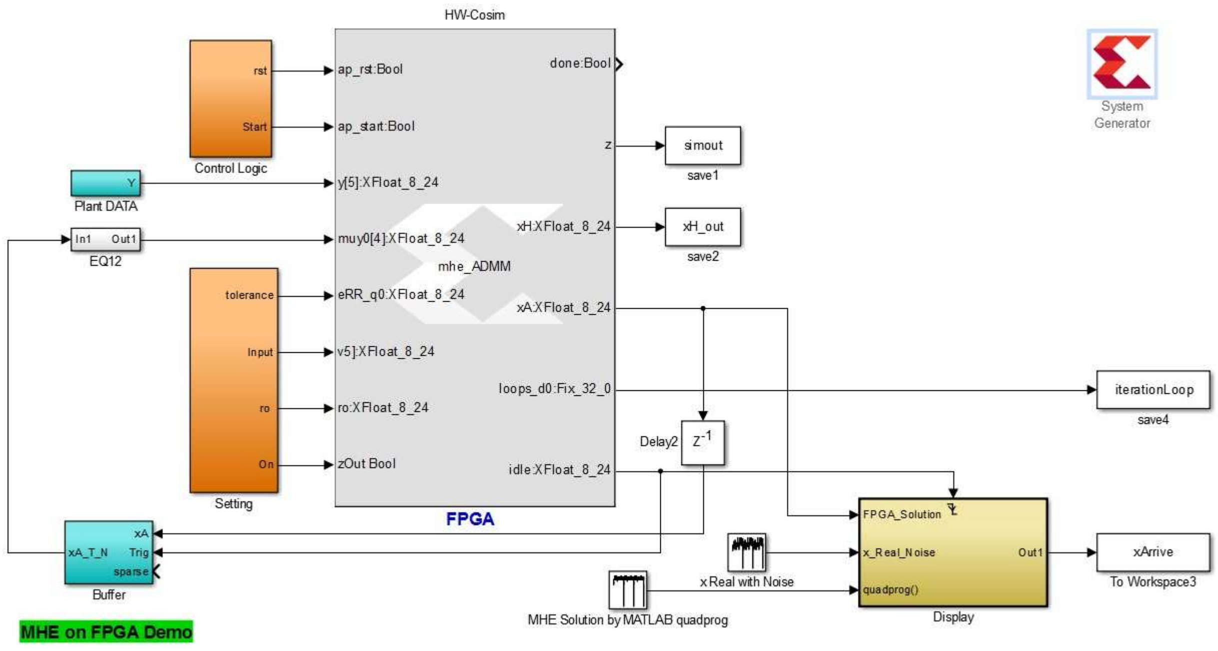


Figure 1: FPGA-in-the-Loop Simulation System

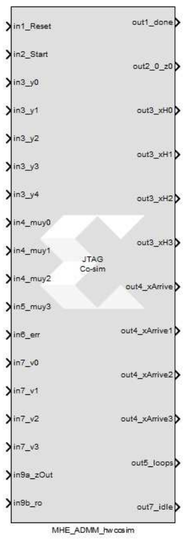


Figure 2: FPGA Harware Co-Simulation Block

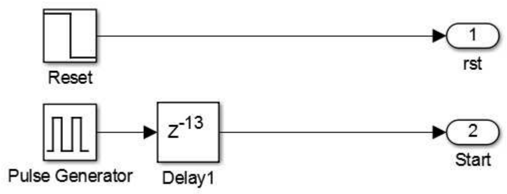


Figure 3: Control Logic Block

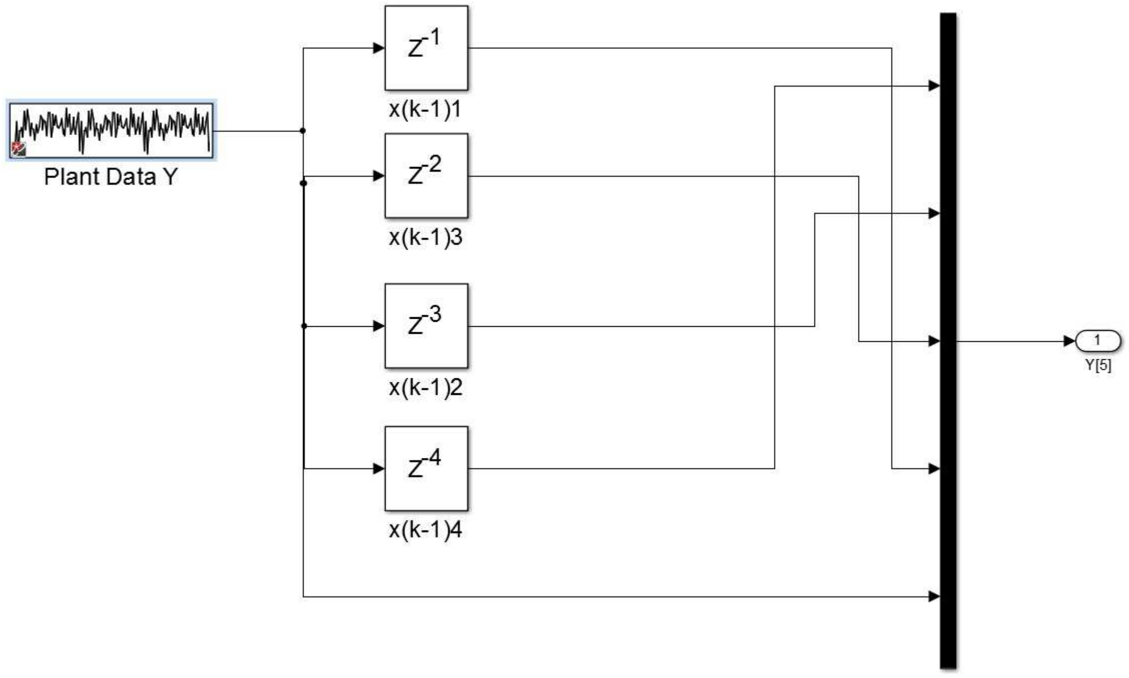


Figure 4: Plant Data Y

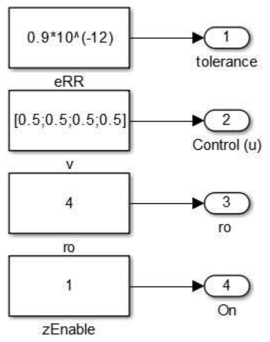


Figure 5: Setting Block

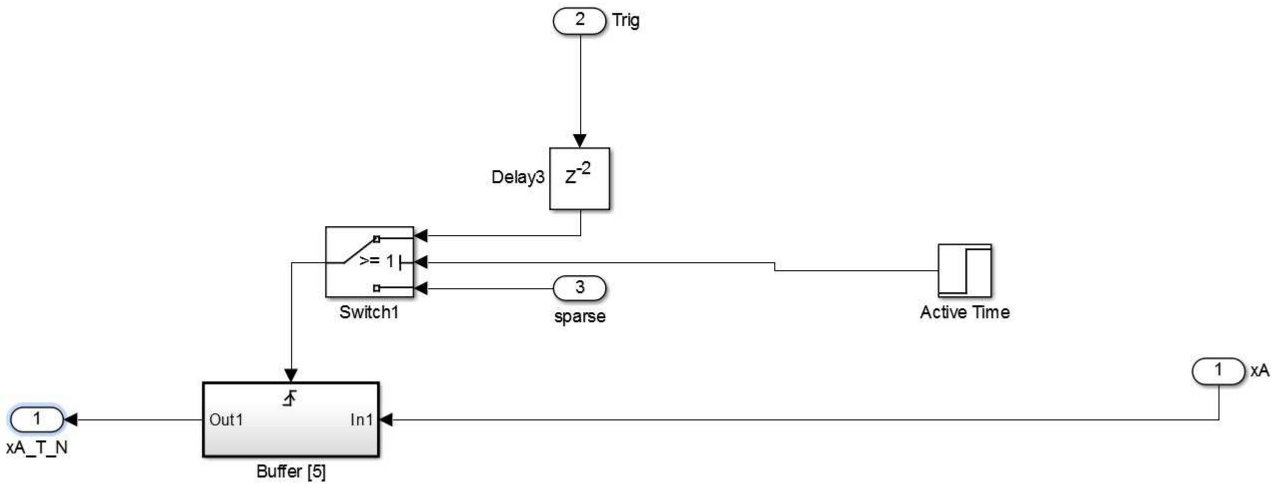


Figure 6: Buffer Block

B "MHE on a Chip" Experimental Setup

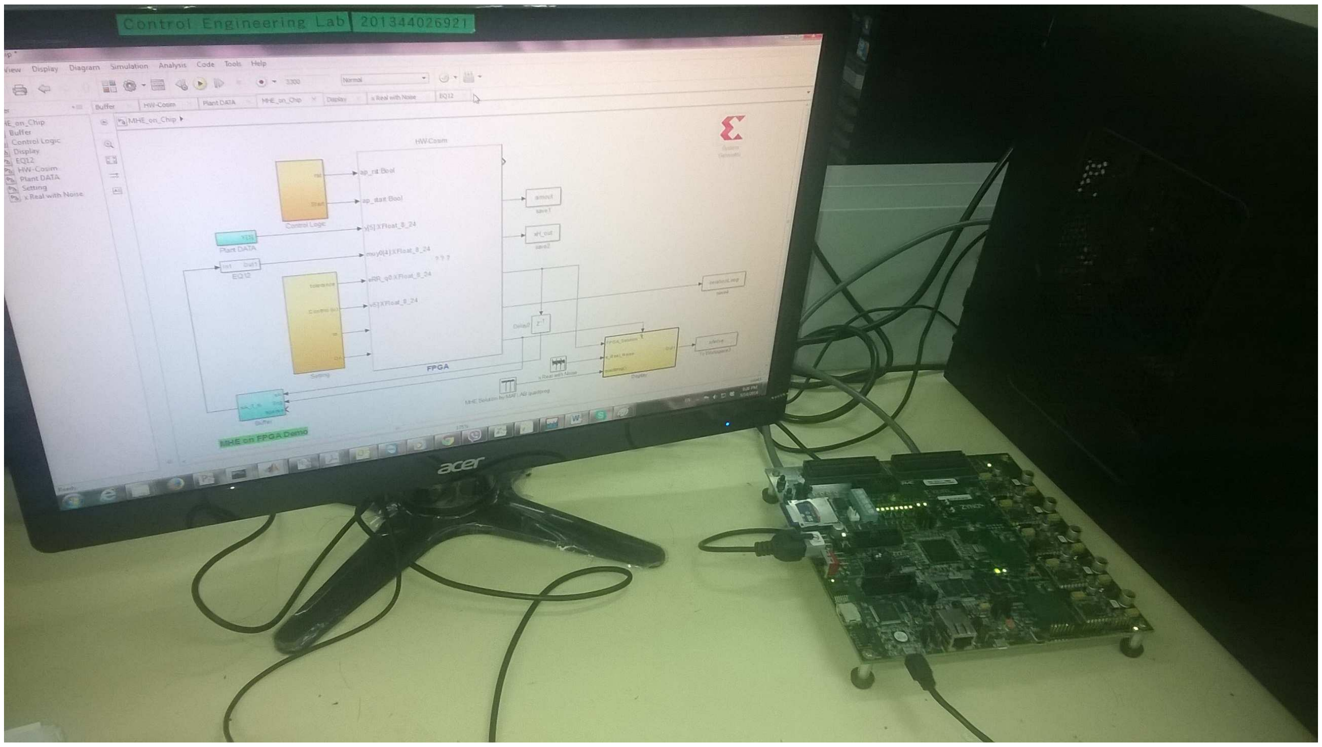


Figure 7: Zynq ZC702 APP SoC

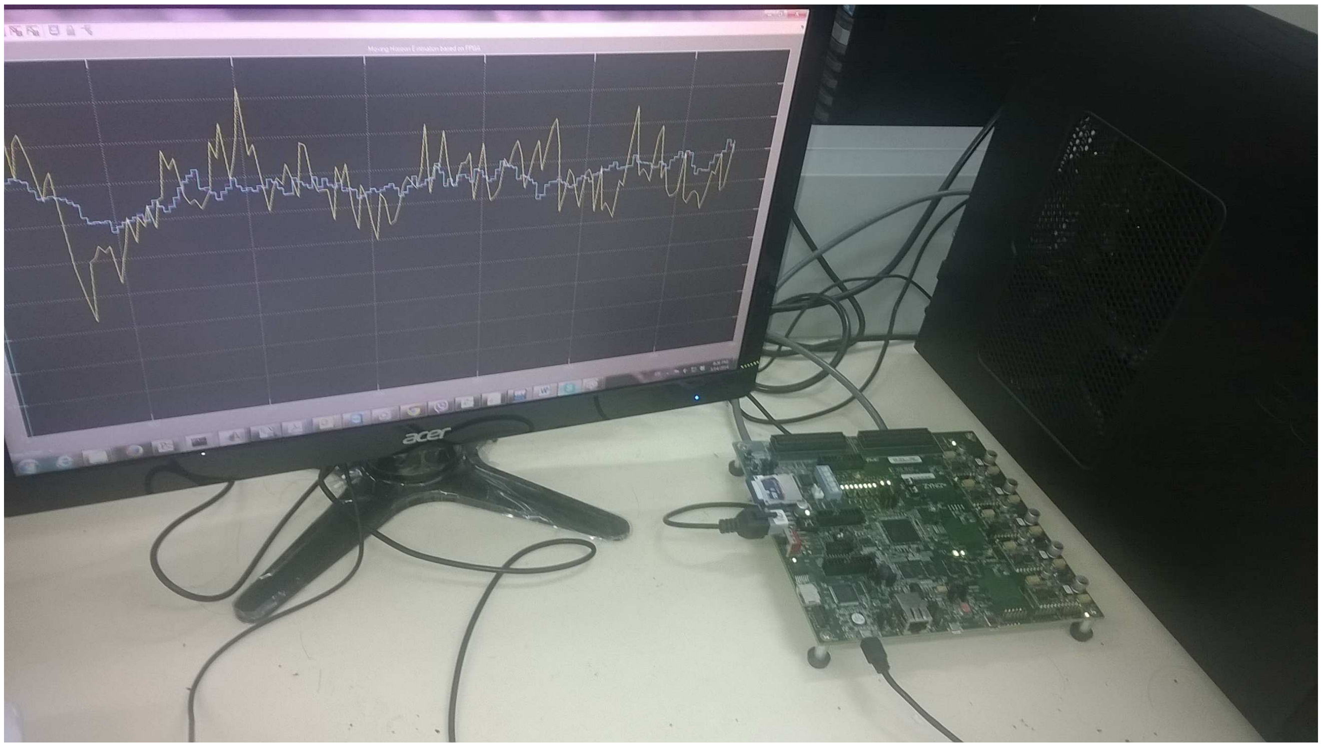


Figure 8: Hardware Co-Simulation

C C++ Code for "MHE on a Chip"

```

C++ Code
/*
* File mhe_admm.cpp
* By DANG Van Thuy @ Control Engineering Lab @ NTU, Singapore
* May 2014
* Ver 1.0
* MHE Project
*/

#include "mhe_admm.h"

//MHE base on ADMM function declaration
int mhe_admm(dType1 y[N],dType1 muy0[n], dType1 TOL[1],dType2 ro[0],dType1 v[N-1], dType1 ...
    z[size],dType1 xH[n],int loops[1],dType1 xArrive[n])
//Input: y, muy0, TOL (Tolerance), ro, v(u)
//Output: z, xH, Loops counter, xArrive
{
dType1 zK[size],zK1[size],uk[size], uk1[size],xK1[size], h[size], g[16], m12g[size];
dType1 epsilon=0;
int i,j,t,loopCounter,count;
dType1 temp;
dType1 zTemp[size];//,thetaTemp[size];
dType2 ro_1;
dType1 epsilon_0;
int loopCount;
int loopNum;
ro_1=0.25;
epsilon_0=TOL[0];

//Variable Initiation
MHE_admm_init:for(i=0;i<size;i++)
{
    zK[i]=zMin[i];
    uk[i]=0;
    xK1[i]=zMin[i];
    zK1[i]=zMin[i];
    uk1[i]=0;
    h[i]=0;
};

// h , g calculation h=-Gama'*Lamda'*Y=-GaLa*[muy0,Y];
MHE_hCalculation:
for(i=0;i<20;i++)
{
    temp=0;
    for(j=0;j<4;j++)
    {
        temp=temp+GaLa[i][j]*muy0[j];
    }

    for(j=4;j<9;j++)
    {
        temp=temp+GaLa[i][j]*y[j-4];
    }
    h[i]=temp;
```

```
}

// g Calculation g=-Bba*v =[-u0 ;-u0 ;u0 -u0....
admmQP_cCalculation:for(i=0;i<4;i++)
{
    g[4*i+1]=-v[i];
    g[4*i+2]=-v[i];
    g[4*i+3]=-v[i];
    g[4*i]=-v[i];
}

//m12g=M12 *g calculation 36x16 16x1
MHE_admm_M12gCalculation:for(i=0;i<size;i++)
{
    temp=0;

loop_label4:for(j=0;j<16;j++)
    {
        temp=temp+M12[i][j]*g[j];
    }
    m12g[i]=temp;
}

//¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬ ADMM loop ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬...
  ¬¬¬¬¬¬¬¬
loopCounter=0;
epsilon=1;

ADMM_loop:do
{
//xK1=M11*(-h+ro*(zK-uk))+M12*g
Loop_1:for(i=0;i<size;i++)
    {
    #pragma HLS PIPELINE    //Loop Pipelining Directive

    temp=0;
Loop_1.1:for(j=0;j<size;j++)
        {
        temp=temp+M11[i][j]*(-h[j]+ro[0]*(zK[j]-uk[j]));
        }
        xK1[i]=temp+m12g[i];
        zTemp[i]=xK1[i]+ro_1*uk[i];
    }

// Projection
Loop_2:for(i=0;i<size;i++)
    {
    #pragma HLS PIPELINE

if(zTemp[i]<zMax[i])    zK1[i]=zTemp[i];
    if(zTemp[i]>zMax[i])    zK1[i]=zMax[i];
    if(zK1[i]<zMin[i])      zK1[i]=zMin[i];
    }

//uk1=xK1+uk-zK1
epsilon=0;
Loop_3:for(i=0;i<size;i++)
        {
#pragma HLS PIPELINE

        uk1[i]= xK1[i]+uk[i]-zK1[i];
        epsilon=epsilon+(zK1[i]-zK[i])*(zK1[i]-zK[i]);
        zK[i]=zK1[i];
        uk[i]=uk1[i];
        }
loopCounter=loopCounter+1;
}
while(epsilon>epsilon_0);

//----------------------------End ADMM-----------------------------------------------------

//Output
admmQP_outZ:for(i=0;i<size;i++)
{
    z[i]=zK[i];
}
loops[0]=loopCounter;
MHE_admm_outXhat:for(i=0;i<4;i++)
{
    xH[i]=zK[i];
    xArrive[i]=zK[16+i];
}
//ack[0]=1;
Return loops[0];
}




/*
* File mhe_admm.h
* By DANG Van Thuy @ Control Engineering Lab
* NTU, Singapore
* May 2014
* Ver 1.0
* MHE Project
*/

#ifndef _admmQP_H
#define _admmQP_H
#include"ap_fixed.h"
#include"stdio.h"
#include "math.h"

typedef ap_fixed<25,6,AP_RND, AP_SAT>  dType1;  // Fixed point Data Type , 25 bits long, ...
    6 integer bits
typedef ap_fixed<18,8,AP_RND, AP_SAT>  dType2;  // Fixed point Data Type , 18 bits long, ...
    8 integer bits


#define  N  5
#define  M 4
#define  n 4
#define  Nn 20
#define  size 36  // QP Size (2*N-1)*n

#define  wL 16
#define  mNu 12

// MHE base on ADMM function

```

```
int mhe_admm(dType1 y[N],dType1 muy0[n], dType1 TOL[1],dType2 ro[0],dType1 v[N-1], dType1 ...
    z[size],dType1 xH[n],int loops[1],dType1 xArrive[n]);

// M11, M12 pre-computed and loaded M11: 36x36 Matrix, M12: 36x16 Matrix
const  dType1 M11[size][size]=
{
{0.0861026975072449,0.0115724687727949,-0.00127297278375129,-0.000925443208968007,0.019995714317645,..
,...,0.0564792028859685},
};

const dType1 M12[size][wL]=
{
{0.118608904521333,0.13640580586447,0.0147584872209736,0.0192466287930242,0.0574180881243037,...
\\
...,0.20929115959644},
};


const dType1 zMax[size]=
        {
                4.30000000000000,
                4.20000000000000,
                2.70000000000000,
                2.30000000000000,
                4.30000000000000,
                4.20000000000000,
                2.70000000000000,
                2.30000000000000,
                4.30000000000000,
                4.20000000000000,
                2.70000000000000,
                2.30000000000000,
                4.30000000000000,
                4.20000000000000,
                2.70000000000000,
                2.30000000000000,
                4.30000000000000,
                4.20000000000000,
                2.70000000000000,
                2.30000000000000,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,
                1,

        } ;

const dType1 zMin[size]=
{
        2.20000000000000,
        2.40000000000000,
        0,
        0,
        2.20000000000000,
        2.40000000000000,
        0,
        0,
        2.20000000000000,
        2.40000000000000,
        0,
        0,
        2.20000000000000,
        2.40000000000000,
        0,
        0,
        2.20000000000000,
        2.40000000000000,
        0,
        0,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,
        -1,

} ;

const dType1 GaLa[20][9]=
{
        {-6.99897353428822  ,3.00726673635644   ,0.397510911062318  ,0.482987654546387  ...
            ,-0.200000000000000 ,0  ,0  ,0  ,0},
        {3.00726673635644   ,-6.95879175306943  ,0.190294233491864  ,0.691613831146628  ...
            ,-0.100000000000000 ,0  ,0  ,0  ,0},
        {0.397510911062318  ,0.190294233491864  ,-6.88479688368285  ...
            ,0.00358283802374717    ,-0.200000000000000 ,0  ,0  ,0  ,0},
        {0.482987654546387  ,0.691613831146628  ,0.00358283802374717    ...
            ,-6.86404367001155  ,-0.100000000000000 ,0  ,0  ,0  ,0},
        {0, 0   ,0  ,0  ,0  ,-0.200000000000000 ,0  ,0  ,0},
        {0, 0   ,0  ,0  ,0  ,-0.100000000000000 ,0  ,0  ,0},
        {0 ,0   ,0  ,0  ,0  ,-0.200000000000000 ,0  ,0  ,0},
        {0, 0   ,0  ,0  ,0  ,-0.100000000000000 ,0  ,0  ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,-0.200000000000000 ,0  ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,-0.100000000000000 ,0  ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,-0.200000000000000 ,0  ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,-0.100000000000000, 0  ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.200000000000000 ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.100000000000000 ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.200000000000000 ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.100000000000000 ,0},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.200000000000000},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.100000000000000},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.200000000000000},
        {0  ,0  ,0  ,0  ,0  ,0  ,0  ,0  ,-0.100000000000000}

};

#endif
```